Requirements Change Management Guideline

A guideline written by an experienced software development executive describing the importance of and the requirements for implementing comprehensive requirements change management. Building on 20 years of lessons learned, this guideline provides the suggested components, processes, and workflows for a robust change management system. It also argues convincingly the need to include evaluation and communication of the impacts of any change requests.

Change is inevitable. Knowing how to handle changes and requests for changes is vital to delivering the right product on time. Properly estimating the impact of a requested change is one of the most important jobs of the engineering manager, and communicating that impact in a clear way is vital to the success of the project. It's almost embarrassing to ask for an additional 4 hours of budget for a change, and perhaps that's not always necessary. But the cumulative effect of 50 4–hour changes—that's easier to understand and shows why it's necessary to have a process that includes impact evaluation and documentation.

The case illustrations and lessons learned will also be helpful for anyone trying to build a business case for implementing a requirements management system in their organization.

To implement change management successfully, three elements must be defined, implemented, and adopted. This guideline provides an overview and practical examples for all three elements:

- **A Requirements Management System**, for creating and tracking requirement baselines. This can be created in a spreadsheet or in a dedicated requirements management tool.
- **A Change Process**, with no exceptions or circumvention allowed. A typical software change process is illustrated.
- **A Change Tracking System** to keep track of it all. Suggested elements and workflow are provided.

Requirements Change Management

Halfway through coding, the Product Manager requests an additional feature. The developer says it's easy. It is recorded in a Word document, but never included in the official requirements specification. The developer's project work ends up late because of this "easy" feature. Worse, the new feature changes some internal software interfaces; during testing, QA discovers that four features which use those interfaces aren't working as specified. The choice: remove the added feature, or revise four other features to work with it. Either will mean a large schedule slip.

If there's one thing that was consistent throughout my career, it was that as soon as requirements were approved, a change request was in process for some part of the system. It's the nature of engineering work, especially on software projects. Software seems easier to change than hardware, and most of the time that's true. But "easier" doesn't mean "free." Properly estimating the impact of the requested change was one of the most important jobs of the engineering manager, and communicating that impact in a clear way was vital to the success of the project. It's almost embarrassing to ask for an additional 4 hours of budget for a change, and perhaps that's not always necessary. But the cumulative effect of 50 4–hour changes—that's easier to understand and shows why it's necessary to have a process that includes impact evaluation and documentation.

Many times the "simple" change that was being requested was not well thought out, and resulted in either an implementation that was not what was intended or more work than originally believed. Sometimes the developer started work on a change before it was approved, only to discover that it never was.

This guideline captures lessons learned during my career about the importance of actively managing the change process, and of building impact evaluation and impact communication into that process.

Intended Audience:

- **VP Engineering, SW/QA Directors, and Project Managers** who want to keep their projects on schedule and avoid surprises due to unexamined change impacts
- **Development Managers and Technical Leads** who want to ensure that their developers are working on approved features to a schedule that is understood by everyone
- **Developers** who want to ensure that they are building the right product
- **Heads of PMOs or Project Sponsors** who want to feel confident that the anticipated product will be released as promised

OVERVIEW

Has anyone *ever* worked on a project that was delivered exactly as originally specified, with no changes whatsoever? When requirements change, who analyzes how the greater complexity might affect existing development or project end dates? Who approves the changes, and how do developers know when approval has been given?

As development continues, it is certain that requirements will need to be clarified or corrected. It is inevitable that some requirements will change, and just as inevitable that additional requirements will be requested. A well-documented set of requirements forms the basis for changes and additions and allows impact estimates to be made, given, and accepted with confidence. The management of this requirements baseline and how it is used in the change management process is documented in this guideline.

Requirements engineering involves a requirements capture process followed by a well defined and enforced change management process, and includes a traceability process. This guideline describes the second component: requirements change management.

Requirement specifications change because:

- Requirements were unclear and need to be clarified
- The customer reverses a decision or wants something added to the product
- Marketing needs a change or addition to better market the product
- The schedule is slipping, and simplification is possible
- A feature can't be implemented as specified or as designed

A typical change management nightmare goes something like this. Halfway through development, a change request comes in from the customer. The Product Manager wants to please them, and everyone thinks the change is an improvement. The developers are told to go ahead, and the requirements document is even revised to reflect the new requirement. But much of the existing code is now obsolete, and the new code is more complicated to implement. The product schedule is not updated because no one was asked to evaluate the change's impact. The schedule slips, the change is blamed, and the customer says, "If we had known that that change would have caused a 6-month slip in the project, we never would have requested it."

Change is inevitable. Knowing how to handle changes and requests for changes is vital to delivering the right product on time. To implement change management, three elements must be defined, implemented, and adopted:

- A Requirements Management System
- A Change Process

- A Change Tracking System
- Each of these is discussed below.

REQUIREMENTS MANAGEMENT SYSTEM

Knowing your requirement base when development begins is key to the change process. Only then can you start making and tracking changes.

Marketing people like to think of requirements in terms of entire features, but within each feature are many low-level requirements that must be developed and tested. In this guideline, we will call these "requirement objects." Individual requirement objects may change without greatly affecting what is considered the "feature."

For example, a Reporting feature might describe how a user requests a report, how the user knows when the report has completed, and how the user views the report results. Suppose a change is made to the notification method (e.g. using an email instead of an alert). A requirement object must be changed, but the basic Reporting feature is not greatly altered.

It might appear that since no one seems to care whether notification is via email or an alert it doesn't need to be documented as a requirement. But the developer cares and needs guidance, the tester cares and needs to understand how to test the feature, and product consistency might matter to someone eventually.

To effectively track changes, it is important to have a method of tracing the change to only the affected requirement object(s). To do this, it is critical to have the requirements documented in a Requirements Management (RM) system that distinguishes between each requirement object and allows attributes to be associated with each one individually. Change tracking features in word processors are not particularly suited to this, but spreadsheets can be used successfully, and there are a variety of dedicated Requirements Management tools. A requirements management tool has the added advantages of automatic assignment of unique IDs for each requirement object and implementing change history tracking down to the individual object. See the INCOSE Requirements Management Tools Survey at http://www.paper-review.com/tools/rms/read.php for a thorough comparison of many Requirements Management tools.

Regardless of what tool is used, it is important to define the attributes to associate with each requirement object.

Critical Attributes for a Requirement Object

Attribute	Use
Requirement ID (should be unique within the entire system)	Uniquely defines the requirement so it can be referenced unambiguously

Attribute	Use
Requirement text	Exactly specifies a feature that must be developed and also serves as the basis for testing
Version or Release	Specifies the implementation version, allowing: An understanding of what is in the current release Inclusion of requirements in future releases A reminder of which release implemented or changed a requirement
Change Requirement ID (CR ID)	References each change in the change tracking system that changed (or created) this requirement, by ID code or number.

Traceability might require us to expand these attributes, but Requirement ID, Requirement text, Release, and CR ID are sufficient for managing change. These attributes provide for requirement baselines and change tracking.

Using a spreadsheet, the management system might look like this.

	A	B	C	D
1	**Req ID**	**Requirement Text**	**Release**	**CR ID**
2	CAT-1	1 Catalog Module		
3	CAT-2	1.1 Licensing Options		
4	CAT-3	If the product catalog is licensed, then there will be a Catalog Tab available for all users.	1.0	
5	CAT-4	and all users will have related Product Catalog capabilities available in the User Capabilities list.	1.0	
6	CAT-110	The "Manage Parts" radio button will also be available for users with that capability.	2.0	12345 12378
7	CAT-7	If Catalog is licensed, then users can use the basic parametric search parameters.	1.0	
8	CAT-8	If the Catalog is not licensed, then only the Select a part function will be available on transaction screens,	1.0	
9	CAT-9	and there will be no catalog tab or related capabilities.	1.0	
10				

Figure 1 Example of a requirements management system implemented as a simple spreadsheet

A dedicated requirements management tool—whether commercial or open source—usually has several important advantages over a simple spreadsheet solution like the one above:

- Automatic creation of the Requirement ID, ensuring that each will be unique
- History of requirement changes: who, what, when, and how

- Baseline control
- Access control (who is allowed to modify)
- Automated parsing of Word or other text documents, so requirements can be written and initially reviewed in a format that is easily read
- Inclusion of graphics in the requirement text
- Automatic links to the change request (if in a compatible tool)
- Links to other files, such as higher-level specifications, test plans, reports, etc.
- Complex filtering based on attribute values

CHANGE PROCESS

As always, it's important to start with a process and follow it. In the case of a software project, the Requirement Change process can be similar to the Code Change or Bug Tracking process, and many companies combine the two, distinguishing in who may initiate, review, and authorize, and in some of the process flow rules. A combined process is usually called the Software Change process. This process is usually implemented with a Software Change Review Board, or SCRB, that is composed of representatives of the stakeholders, typically:

- Quality Assurance (QA) / Test – can act as chair and recorder
- Engineering (Director, VP, and/or CTO)
- Marketing
- Client Technical Support

Example Requirements Change Process

Process Step	Parties involved
Document the requirement revision(s) or the new requirement(s), including the rationale for the change (especially if work has already begun or if it alters something already developed).	Requestor can be anyone, but is typically someone in: • Marketing – usually to add functionality or clarify a requirement based on a question from Engineering • Engineering – usually to change something that is overly complicated or to clarify a requirement
Create a change request that includes this documentation.	Requestor
If this change is merely clarification of an existing requirement and there is no schedule impact:	
Approve the request.	SCRB or Change Review Board
Revise the requirements baseline.	QA or whoever is authorized
Notify the developer (who is probably already part of the process).	Engineering representative
Otherwise (if there is a real change):	
Estimate the work involved in this change (both development and test), the impact on other parts of the system, and the impact on the overall schedule.	Engineering
Create a new schedule estimate.	Engineering management
Approve (or deny) the request. If approved:	SCRB or Change Review Board
• Change the requirements baseline.	QA or whoever is authorized
• Make sure developer knows of and understands change.	Engineering management
• Make sure all stakeholders know of schedule impact based on the clarification or new requirement.	SCRB or Change Review Board

Process Step	Parties involved
• Make sure QA bases testing on the documented and approved requirements that include all changes and additions.	QA

Negotiating Changes

If any change or addition cannot be handled without changing end date, and if that is not acceptable to Marketing (which is normally the case), you will need to decide what to trade off in order to avoid impacting the end date. This might mean:

- adding staff to implement this or another feature/requirement (increasing cost).
- shuffling assignments to use someone who otherwise would have finished earlier (increasing cost).
- removing another, less-important requirement/feature (trading off features).
- simplifying the request so it is more easily implemented (trading off features).
- delaying all or part of the change until a later release (trading off features).

Think of the project schedule, cost, and feature set as a triangle.

If any one of these items change, at least one of the others must somehow change.

- If there is a request for added scope (a new feature or expansion of an existing feature), then unless another feature is traded off or simplified, either the schedule must change or the cost (staffing) must change.
- If an accelerated release schedule is requested, then either the cost must increase (more staff), or the features must be reduced.
- If a cost reduction is requested (typically this means fewer staff, but it could mean less-experienced staff), then the features should be reduced (extending the schedule with fewer staff may or may not reduce total cost).

CHANGE REQUEST TRACKING SYSTEM

Change Requests are tracked using a tracking system that allows a Change Request (CR) to be opened, reviewed, acted upon (authorized or denied), and eventually closed. The change tracking system requires several components.

A Workflow Definition, which describes:

- the possible states for the CR (e.g. Open, Resolved, etc.)

- the roles of system users (who can file a CR, who can change the state of a CR, who can be assigned a CR and under what circumstances, who can close a CR, who can view a CR)

- how the CR flows from one state to another (who can move the CR from one state to another, which flows are possible, which ones are typical, which ones are disallowed)

Data items that define the request, such as:

- CR ID

- title

- submittal date

- requestor name

- description of problem or modification requested, including rationale

- type of CR (bug, enhancement, change, etc.)

- release or version that the problem was found in

- how to reproduce it, if it is a perceived bug

- requirement ID(s) affected or not being met

- release or version where the fix/change should be made

- severity of the problem or issue

- priority of the fix or change

Data items that define the state of the CR and other data filled in by those assigned to do so, such as:

- current state of the CR (e.g., new, open/evaluate, open/fix, resolved, closed)

- current CR assignee

- person(s) to be notified when the CR changes state or when a comment is added

- person assigned (to be assigned or was assigned) as developer

- estimated or actual development hours

- person assigned (to be assigned or was assigned) as tester

- estimated or actual test hours
- assigned code reviewer, or a status indicating the code was reviewed (and possibly, date)
- resolution date (date code was fixed/changed)
- resolution description, including files fixed/changed
- date of test completion

Example Software Change Request Workflow

Action	CR State	CR Assignee
Change Request submitted by Requestor	New	SCRB Chair
Review Change Request at SCRB Meeting	Open-Estimate	Eng. Mgr.
Estimate new work and record as a comment in the Change Request	Open-Estimate	SCRB Chair
SCRB approves change	Open-Fix	Eng. Mgr.
Engineering Manager updates specification with change	Open-Fix	Developer
Developer makes the change to the code	Resolved	QA
QA tests change	Closed	

You can see how this process could be used for tracking anything. It often forms the basis of an "issues" tracking system or includes "issues" as a different type of item to track (as opposed to bugs or software enhancements). It is crucial, however, that the change process includes and is accepted by all organizations.

It is also important that the authority be defined within this process. For example, in the action "SCRB approves change," who really has the ability to authorize a change request? Hopefully it's someone who understands what is required of the product, like the Product Manager. But who can authorize a change that would result in a longer schedule? That might be the CEO, a Vice President, or a Division President.

Administrative Information

Revision	Author	Date	Sections Affected	Change Summary
1.0		1/3/2009		

Current Version	1.0
Date	1/3/2009
Master Document Chapter Number	1
Document ID	110